Introduction to Hibernate

What is Hibernate?

Hibernate is an open source object relational mapping (**ORM**) tool that provides a framework to map object-oriented domain models to relational databases for web applications.

Object relational mapping is based on
the containerization of objects and the abstraction that provides that capacity. Abstraction makes it possible to address, access and manipulate objects without having to consider how they are related to their data sources.

The Hibernate ORM framework guides **mapping Java classes** to database tables and Java data types to SQL data types
and provides **querying and retrieval**.

How does Hibernate work?

Hibernate is an open source Object-Relational Persistence and Query service for any Java Application. Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieves the developer from most common data persistence related programming tasks.

Hibernate sits between traditional Java objects and database server to handle all the works in persisting those objects based on the appropriate O/R mechanisms and patterns.

Why we use Hibernate?

Hibernate eliminates the shortcomings of other technologies like **JDBC**. Let's take a look at how it optimizes the tasks better than JDBC.

- Hibernate overcomes the database dependency faced in the JDBC.
- Changing of the databases cost a lot working on JDBC, hibernate overcomes this problem with flying colors.
- Code portability is not an option while working on JDBC, which is handled by hibernate easily.
- Hibernate strengthens the object level relationship.
- It overcomes the **exception-handling** part which is mandatory while working on JDBC.
- Hibernate overcomes the object level relationship.
- It reduces the length of code with increased readability by overcoming the boiler plate problem.

Hibernate provides optimal and efficient solutions for any task by overcoming all the shortcomings of JDBC. Let us take a look at various operations along with technologies and databases we can work on while using the hibernate framework in Java.

Functionalities Supported By Hibernate

- Hibernate uses Hibernate Query Language which makes it database independent.

- It supports auto DDL operations.
- Hibernate has Auto Primary Key Generation support.
- It supports Cache memory.
- Exception handling is not mandatory for hibernate.
- The most important is hibernate is an ORM tool.

Advantages of Hibernate over JDBC –

1. Developer has to write code in JDBC to map an object model's data representation to a relational data model and its corresponding database schema.

2. Hibernate itself takes care of this mapping using XML files so developer does not need to write code for this.
   JDBC supports only native Structured Query Language (SQL).
   Hibernate provides Hibernate Query Language (HQL) which is similar to SQL syntax and supports polymorphic queries too. It also supports native SQL statements.

3. The mapping of Java objects with database tables has to be taken care of in JDBC. Hibernate provides transparent persistence and therefore there is no need to map database tables tuples to application objects during interaction with RDBMS.

4. With JDBC, caching needs to be manually maintained. Hibernate cache is set to application work space. Relational tuples are moved to this cache as a result of query. It improves performance during multiple writes for the same data.

5. In JDBC there is no check that always every user has updated data. Hibernate enables definition of version type field to application, due to which Hibernate updates version field of database table every time relational tuple is updated in form of Java class object to that table.